

# Documentation Generator Focusing on Symbols for the HTML-ized Mizar Library\*

Kazuhisa Nakasho<sup>1</sup> and Yasunari Shidama<sup>2</sup>

<sup>1</sup> Shinshu University, Japan, [13st205f@shinshu-u.ac.jp](mailto:13st205f@shinshu-u.ac.jp)

<sup>2</sup> Shinshu University, Japan, [shidama@cs.shinshu-u.ac.jp](mailto:shidama@cs.shinshu-u.ac.jp)

**Abstract.** The purpose of this project is to collect symbol information in the Mizar Mathematical Library and manipulate it into practical and organized documentation. Inspired by the MathWiki project and API reference systems for computer programs, we developed a documentation generator focusing on symbols for the HTML-ized Mizar library. The system has several helpful features, including a symbol list, incremental search, and a referrer list. It targets those who use proof assistance systems, the volume of whose libraries has been rapidly increasing year by year.

**Keywords:** Mizar, mathematical knowledge management, search system, documentation generator

## 1 Motivation

In mathematical knowledge management (MKM), expanding of the fields covered by formal methods has led to the rapid growth of formal mathematical libraries. For instance, the Mizar Mathematical Library (MML)[6,8,9] has grown to more than 2.7 million lines in 2015, and it has been increasing by approximately 0.1 million lines per year.

The development of formal mathematical libraries facilitates the reuse of mathematical symbols and theorems, thereby improving the efficiency of writing formal proofs. However, the increased volume of the libraries makes it difficult for users to grasp what and where symbols and theorems are defined. In recent years, developers of formal proofs have spent considerable time on search tasks in large-scale libraries, thereby decreasing the productivity of formal verification. Therefore, searching and browsing efficiency in large-scale libraries has been a crucial issue in MKM.

## 2 Survey and Design Decision

We analyze some existing tools for searching and browsing the Mizar library. The HTML-ized Mizar library[1,15] is one of the most successful documentation tools

---

\* The final publication is available at <http://link.springer.com>.

for formal mathematical libraries. The HTML-linked MML was first developed in the late 1990s for the former "Journal of Formalized Mathematics" <sup>3</sup> and then re-implemented by Dr. Josef Urban using the XML/XSLT technology. This new HTML-ization was also used in the MathWiki Project.<sup>4</sup> The system is capable of intuitive and rapid browsing as a result of hyperlinks being embedded into symbols, enabling users to jump from symbol occurrences to their definitions by clicking them. The system has been widely used by Mizar users because of its effectiveness and user-friendly design. However, because this system does not have retrieval functions, users are frequently obliged to grep symbols in the MML using text editors. Moreover, although the hyperlinks allow users to jump to their definitions, it is still difficult to, inversely, enumerate the symbols that include a particular symbol in their definitions.

MML Query[4,5] is the most flexible and sophisticated search system for the MML. This system has its own query language, and users can input more detailed information regarding search objects than is possible using grep. However, users must learn and master the query language, thus this is a burden for beginners.

Conversely, in software development, most widely used programming languages have several types of API documentation generators, and almost all of the widely used libraries have their own online API documentation systems. Those API reference systems have common features, such as incremental search and a list of symbols that is automatically created by API documentation generators during library updates. Many documentation generators, such as Doxygen<sup>5</sup> and RDoc<sup>6</sup>, have contributed to the acceleration of software development.

We apply the software development approach to developing a documentation generator that works on the MML in order to overcome the drawback of existing search and browsing systems.

### 3 Application

Using the programming language Python,<sup>7</sup> we developed a documentation generator<sup>8</sup> that comprises the following three steps:

1. Parse the HTML-ized MML and collect symbols and their mutual relationships.
2. Clean and arrange those data.
3. Output reference documents in HTML format. Each file corresponds to one symbol.

These steps take only a few minutes in total.<sup>9</sup>

<sup>3</sup> <http://mizar.org/JFM>

<sup>4</sup> <http://www.ru.nl/foundations/research/projects/mathwiki/>

<sup>5</sup> <http://www.doxygen.org/>

<sup>6</sup> <https://github.com/rdoc/rdoc>

<sup>7</sup> <https://www.python.org/>

<sup>8</sup> <https://github.com/aabaa/mmlfrontend>

<sup>9</sup> Windows 7, CPU: AMD A10-5800K 3.8 GHz (4-core), Memory: 16.0 GB

The latest reference system produced by the generator is available at a website.<sup>10</sup> Fig. 1 shows a screenshot of the system.

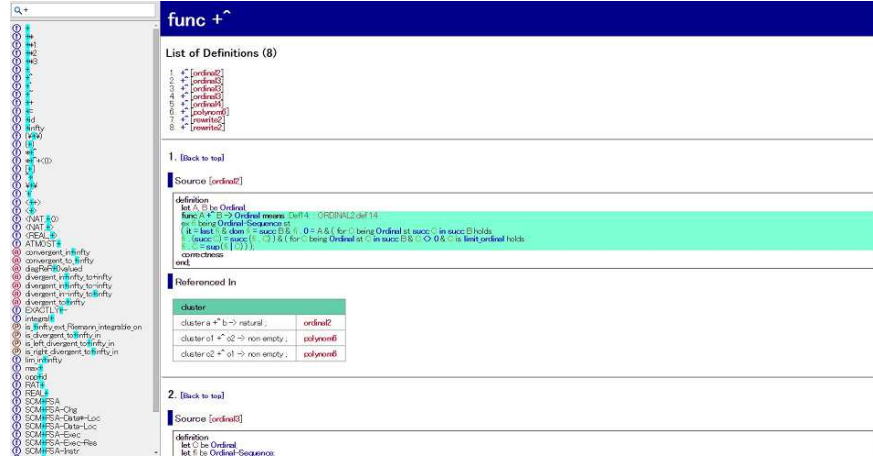


Fig. 1. Screenshot of the reference system.

The reference system offers the following helpful features:

**Symbol List:** There are nearly 9,000 symbols (predicate, mode, structure, functor, and attribute) in the MML, all of which are listed in the left pane of the system. The type of each symbol can be distinguished by the icon next to the symbol. Clicking a symbol in the list causes the corresponding page to be loaded into the main frame in left pane.

**Incremental Search:** An incremental search function is located at the top of the left pane. When several search words separated by blanks are input, the system combines the symbol list into symbols that contain all of the indicated words. As the system has an original search table, the function returns search results immediately. Users can quickly look up symbols defined in the MML, even without knowing the correct spelling.

**Source Code:** The symbol definition source code is imported from the HTML-ized MML. Symbols in bold font are hyperlinked to their definitions. Internal links pointing to their definitions in this reference system are in blue. External links pointing to their definitions in the original HTML-ized MML are in red.

**Referrer List:** Although the HTML-ized MML enables users to jump from symbol occurrences to their definitions by clicking them, it does not have a function to enumerate symbols that are used in the definitions of particular symbols. The new system organizes the list of referrers for each symbol, and users can check them easily.

<sup>10</sup> <http://webmizar.cs.shinshu-u.ac.jp/mmlfe/current/>

## 4 Conclusion and Future Work

We utilized the API documentation technique from the field of software development to develop a new documentation generator that works on the MML. This system enables users to retrieve symbols quickly and intuitively using an incremental search function. Furthermore, users can easily check the types of symbols allowed to be used together by referrer lists. These functions have contributed considerably to improving the efficiency of formal proof development, and the system has gained a good reputation among the Mizar community. Additionally, the approach of the system is not specific to Mizar and the MML, thus all formal libraries would benefit from such a system. Therefore, the future versions of the system should support other formal languages and libraries.

We mention three remaining issues regarding the system:

**Reimplementation with the XML-ized MML:** The current documentation generator parses the HTML-ized MML instead of the XML-ized Mizar[12]. This is because the former represents relationships between symbols and their definitions as embedded hyperlinks, whereas it is difficult to collect these relationships from the latter. However, the extra process required to generate the HTML-ized MML takes considerable time. Therefore, we would like to change the system to work with the XML-ized MML in the future.

**Theorem Search:** A theorem search system requires semantic analysis, and machine learning would be a promising approach. Because this research is underway for automated reasoning[3,14], we would like to apply the technique to an interactive search engine.

**Tagged Comments:** In software development, most documentation generators collect tagged comments, such as authors, purposes, and usages, and reflect them in API documents, whereas the current Mizar library does not have any tagged comments. Although Mizar is a comparatively readable formal language, it is sometimes difficult to discern a writer's intention from a source code. Consequently, such a function would work beneficially, if it were implemented. Furthermore, there is no standard for tagged comments in formal libraries, such a format should be developed in future work and then adopted by all formal libraries.

We also suggest a possible application of the system:

**Code Completion:** Other major proof assistants have developed graphical interfaces, such as the jEdit plugins for Coq and Isabelle [10,17,18]. Although the Mizar system provides an Emacs plugin[13], some users hope that a newer one will be offered on a modern integrated development environment (IDE). The incremental function of the system would assist in implementation of code completion for those IDE systems.

## 5 Acknowledgment

The authors wish to thank the members of the MathWiki Project for their preceding work. Our research is deeply dependent on their product. We especially

express our gratitude to Dr. Josef Urban, who is well known as a member of the MathWiki Project, for giving us some beneficial advice for the study. We would also like to thank to Dr. Adam Naumowicz for helping us improve our paper.

## References

1. Jesse Alama, Kasper Brink, Lionel Mamane, and Josef Urban: Large Formal Wikis: Issues and Solutions. *Calculemus/MKM 2011. LNCS*, vol. 6824, pp. 133–148. Springer (2011).
2. Jesse Alama, Michael Kohlhase, Lionel Mamane, Adam Naumowicz, Piotr Rudnicki, and Josef Urban: Licensing the Mizar Mathematical Library. *Calculemus/MKM 2011. LNCS*, vol. 6824, pp. 149–163. Springer (2011).
3. Jesse Alama, Tom Heskes, Daniel Kuhlwein, Evgeni Tsivtsivadze, and Josef Urban: Premise Selection for Mathematics by Corpus Analysis and Kernel Methods. *J. Autom. Reasoning* 52(2): 191–213 (2014).
4. Grzegorz Bancerek and Piotr Rudnicki: Information Retrieval in MML. *MKM 2003. LNCS*, vol. 2594, pp. 119–132. Springer (2003).
5. Grzegorz Bancerek and Josef Urban: Integrated Semantic Browsing of the Mizar Mathematical Library for Authoring Mizar Articles. *MKM 2004. LNCS*, vol. 3119, pp. 44–57. Springer (2004).
6. Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz: Mizar in a Nutshell. *J. Formalized Reasoning* 3(2): 153–245 (2010).
7. Artur Kornilowicz: On Rewriting Rules in Mizar. *J. Autom. Reasoning* 50(2): 203–210 (2013).
8. Roman Matuszewski and Piotr Rudnicki: Mizar: the first 30 years. *Mechanized Mathematics and Its Applications* 4(1): 3–24, (2005).
9. Adam Naumowicz and Artur Kornilowicz: A Brief Overview of Mizar. *TPHOLs 2009. LNCS*, vol. 5674, pp. 67–72. Springer (2009).
10. Carst Tankink: PIDE for Asynchronous Interaction with Coq. *UITP 2014*: 73–83.
11. Andrzej Trybulec, Artur Kornilowicz, Adam Naumowicz, and Krystyna Kuperberg: Formal Mathematics for Mathematicians - Foreward to the Special Issue. *J. Autom. Reasoning* 50(2): 119–121 (2013).
12. Josef Urban: XML-izing Mizar: Making Semantic Processing and Presentation of MML Easy. *MKM 2005. LNCS*, vol. 3863, pp. 346–360. Springer (2006).
13. Josef Urban: MizarMode - an integrated proof assistance tool for the Mizar way of formalizing mathematics. *J. Applied Logic* 4(4): 414–427 (2006).
14. Josef Urban: Momm - Fast Interreduction and Retrieval in Large Libraries of Formalized Mathematics. *International Journal on Artificial Intelligence Tools* 15(1): 109–130 (2006).
15. Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers: A Wiki for Mizar: Motivation, Considerations, and Initial Prototype. *AISC/MKM/Calculemus 2010. LNCS*, vol. 6167, pp. 455–469. Springer (2010).
16. Josef Urban, Piotr Rudnicki, and Geoff Sutcliffe: ATP and Presentation Service for Mizar Formalizations. *CoRR abs/1109.0616* (2011).
17. Makarius Wenzel: Isabelle/jEdit - A Prover IDE within the PIDE Framework. *AISC/MKM/Calculemus 2012. LNCS*, vol. 7362, pp. 468–471. Springer (2012).
18. Makarius Wenzel: PIDE as front-end technology for Coq. *CoRR abs/1304.6626* (2013).